

Nonlinear blind source separation for chemical sensor arrays based on a polynomial approximation

Rafael A. Ando
Christian Jutten
Bertrand Rivet
Leonardo Duarte
Romis Attux



UNIVERSITÉ DE
GRENOBLE



gipsa-lab

Grenoble | images | parole | signal | automatique | laboratoire

Overview

1. Problem Statement

- Chemical Sensor Arrays

2. Algorithms and Simulations

- Second-order polynomial for specific case
- Generalized cases

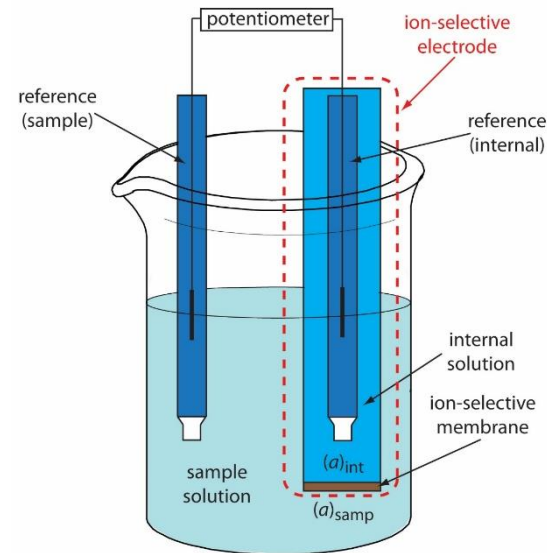
3. Conclusion

Problem Statement

- Description of the problem:
 - $\mathbf{s}(n) = [s_1(n), s_2(n), \dots, s_N(n)]^T$
 - $\mathbf{x}(n) = [x_1(n), x_2(n), \dots, x_N(n)]^T = \mathcal{F}[\mathbf{s}(n)]$
- Estimate $\mathbf{s}(n)$, given $\mathbf{x}(n)$ and information on the sources and/or mixture process.
- Problem is *blind* if mixing model is unknown.

Problem Statement

- Application to chemical sensor arrays:
 - Ion-selective electrodes (ISE) used to measure concentration of ions.



Problem Statement

- Application to chemical sensor arrays:
 - Concentrations roughly follow Nicolsky-Eisenmann equation:

$$x_i = E_i + \frac{RT}{z_i F} \ln \left(s_i + \sum_{j \neq i} a_{ij} S_j^{z_i/z_j} \right)$$

- z_i is the valence
- Equation is empirical and only approximates actual mixing model

Problem Statement

- Application to chemical sensor arrays:
 - Mixture can be adjusted to become polynomial (in $p_i = s_i^{1/z_i}$):

$$x_i^{modified} = \exp\left((x_i - E_i) \frac{z_i F}{RT}\right) = \sum_{j=1}^N a_{ij} p_j^{z_i}$$

- The modified mixture is polynomial in p_i , with degree $\frac{\max(\mathbf{z})}{\gcd(\mathbf{z})}$.

Algorithms

- For the specific case:
 - 2 sources
 - Both have the same valence
- The modified mixture is linear $\rightarrow$ can be solved by standard methods of BSS.
- However, NE equation is only an approximation...

Algorithms

- We assume the mixture is *almost* linear.
- A second-order polynomial might be able to correct the inaccuracies.

$$\mathbf{y}(n) = \begin{bmatrix} s_1 + a_{12}s_2 + b_{11}s_1^2 + b_{12}s_2^2 + c_1s_1s_2 \\ a_{21}s_1 + s_2 + b_{21}s_1^2 + b_{22}s_2^2 + c_2s_1s_2 \end{bmatrix}$$
$$= \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{s}^{\circ 2} + \mathbf{c} s_1s_2$$

Algorithm: specific case

- Basic idea of the algorithm:
 - Estimate the parameters minimizing mutual information (gradient-based algorithm):

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu E \left\{ \frac{\partial \mathbf{y}^T}{\partial \mathbf{w}} \beta_{\mathbf{y}}(\mathbf{y}) \right\}$$

where

$$\beta_{y_i}(\mathbf{y}) = \left(-\frac{\partial \log p(\mathbf{y})}{\partial y_i} \right) - \left(-\frac{d \log p(y_i)}{dy_i} \right)$$

- The score functions can be calculated via efficient methods as described in [Pham, 2004]



Algorithm: specific case

- The derivative $\frac{\partial \mathbf{y}}{\partial \mathbf{w}}$ can be found from the mixing model supposing $\mathbf{y} = \mathbf{s}$:

$$\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{y}^{\circ 2} + \mathbf{c} y_1 y_2$$

- Using implicit differentiation:

$$\mathbf{0} = \underbrace{\left(\mathbf{A} + 2\mathbf{B} \circ \text{diag}(\mathbf{y}) + \mathbf{c} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \mathbf{y}^T \right)}_{\mathbf{J}_g} \frac{\partial \mathbf{y}}{\partial \mathbf{w}} + \begin{bmatrix} y_2 & y_1^2 & y_2^2 & y_1 y_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_1 & y_1^2 & y_2^2 & y_1 y_2 \end{bmatrix}$$

Algorithm: specific case

- Use a recurrent network to find estimates for the specific set of weights found.
- Network based on Newton-Rhapson root-finding algorithm for nonlinear equation system:

$$\mathcal{G}[\mathbf{s}] = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{s}^{\circ 2} + \mathbf{c}s_1s_2 - \mathbf{x} = \mathbf{0}$$

Algorithm: specific case

- Recurrent network is:

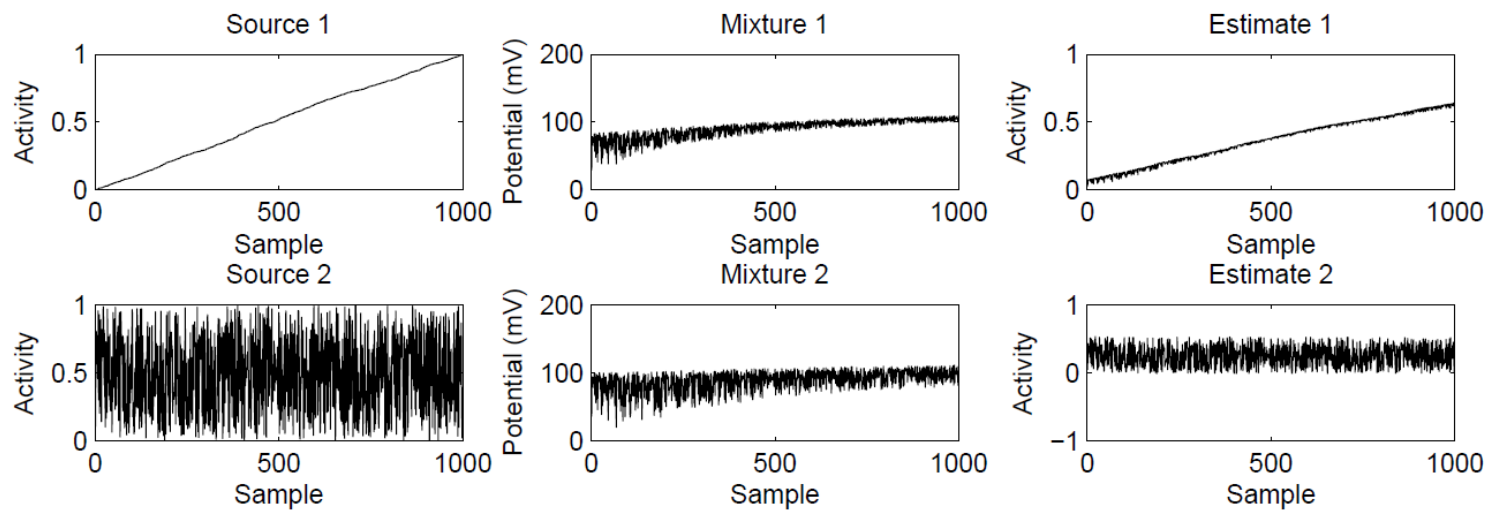
$$\mathbf{y}_{k+1} = \mathbf{y}_k - J_G^{-1}(\mathbf{y}_k)G[\mathbf{y}_k]$$

- J_G is the Jacobian of the mixing model
- Locally stable for all points

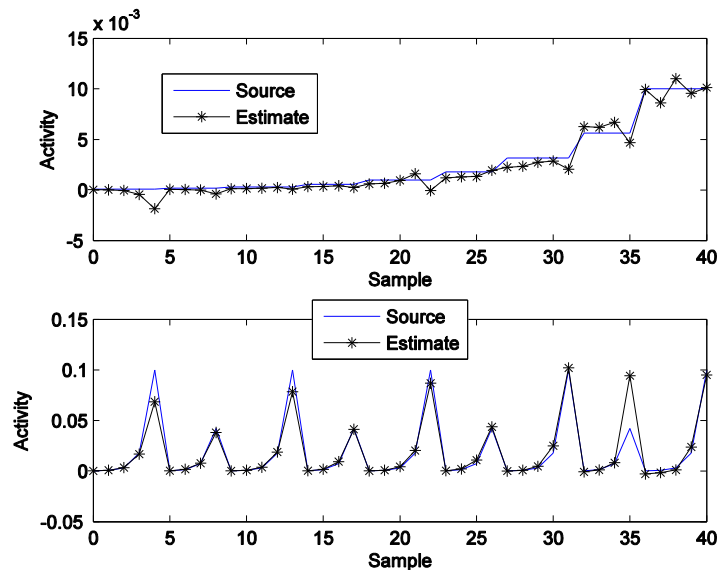
Algorithm: specific case

- Proposed algorithm:
 1. Start with random parameters
 2. Use recurrent network to find $\mathbf{y}$
 3. Use mutual information gradient to update parameters.
 4. Repeat steps 2 and 3 until convergence.

Simulations: specific case



Simulation: specific case



Algorithm	SIR s_1 (dB)	SIR s_2 (dB)
PNL	11.0	10.6
Proposed	16.3	11.1

Compared to the previous algorithm (PNL), the proposed algorithm shows promising results

Algorithm

- Good results, but hypotheses are somewhat limiting.
- A more general version of the algorithm can be performed to allow for:
 - More than 2 sources;
 - Ions with different valences.

Algorithm: general case

- For different valences, we can use a higher degree polynomial instead:
- $z_i = 2 \rightarrow 3^{\text{rd}}$ degree polynomial
- $z_i = 3 \rightarrow 4^{\text{th}}$ degree polynomial
- Example (3^{rd} degree):

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{s}^{\circ 2} + \mathbf{C}\mathbf{s}^{\circ 3} + \mathbf{D} \begin{bmatrix} s_1 s_2 \\ s_1^2 s_2 \\ s_1 s_2^2 \end{bmatrix}$$

- 16 variables!

Algorithm: general case

- For more sources we need more variables.
- Example (3 sources, equal valences):

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{B}\mathbf{s}^{\circ 2} + \mathbf{C} \begin{bmatrix} s_1 s_2 \\ s_1 s_3 \\ s_2 s_3 \end{bmatrix}$$

- 24 variables!

Algorithm: general case

- Gradient of MI can be generalized with the aid of the Kronecker product:

$$A \otimes B = \begin{pmatrix} a_{11} \mathbf{B} & \cdots & a_{1n} \mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1} \mathbf{B} & \cdots & a_{mn} \mathbf{B} \end{pmatrix}$$

- Using implicit differentiation on the mixing model leads to:

$$0 = J_{\mathcal{G}} \frac{\partial \mathbf{y}}{\partial \mathbf{w}} + [\mathbf{I}_n \otimes \mathbf{y}^T | \mathbf{I}_n \otimes \mathbf{y}^{\circ 2 T} | \dots | \mathbf{I}_n \otimes \mathbf{v}_{cross}^T]$$

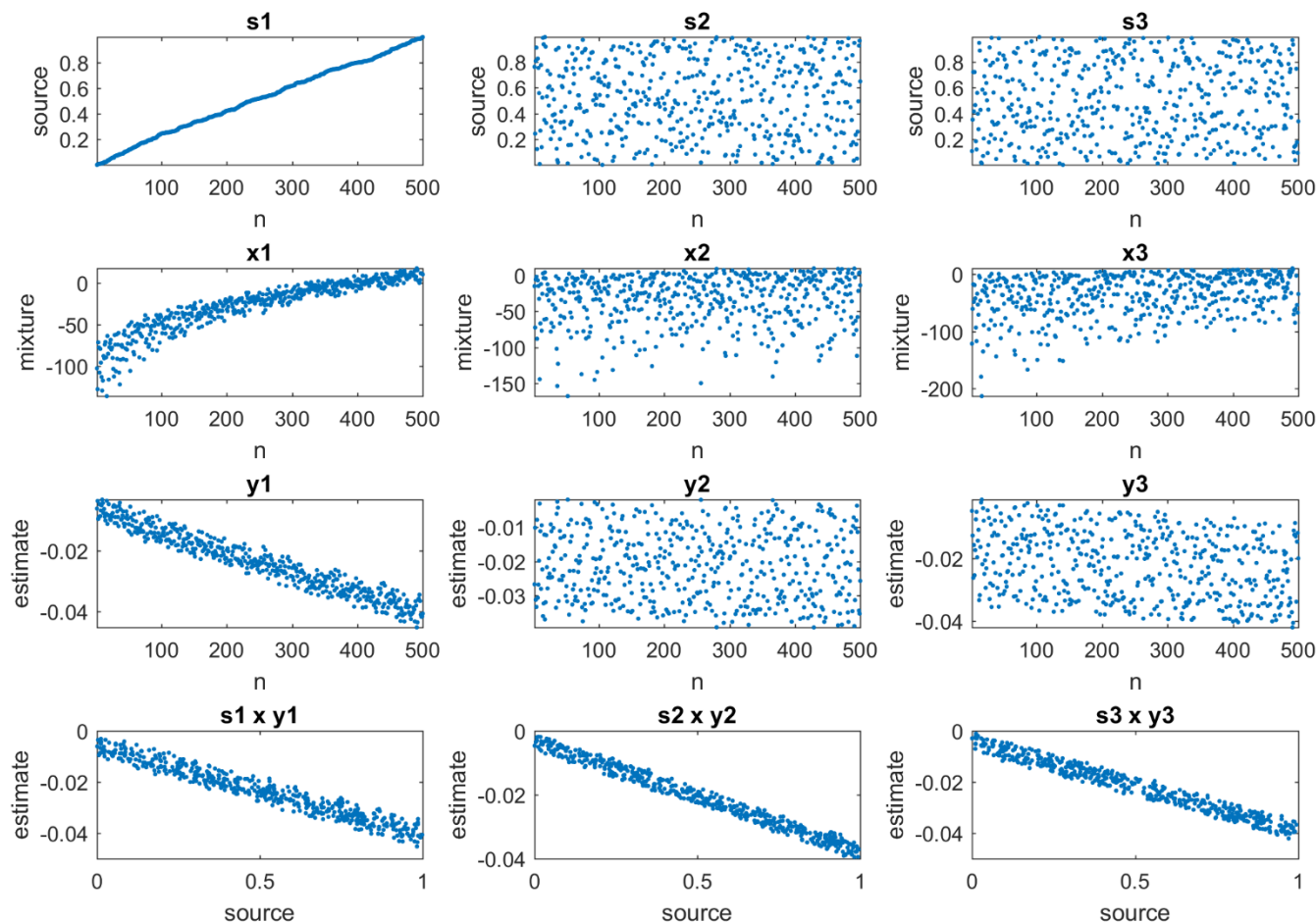
Concatenation

- Works for any number of sources and valences!

Algorithm: general case

- A recurrent network can be developed by the same idea.
- The Jacobian J_G cannot be generalized due to the cross product terms.
- Newton-Rhapson root finding algorithm is not necessarily stable anymore.
- For real chemical data should probably be ok!

Simulation: 3 sources, equal valence



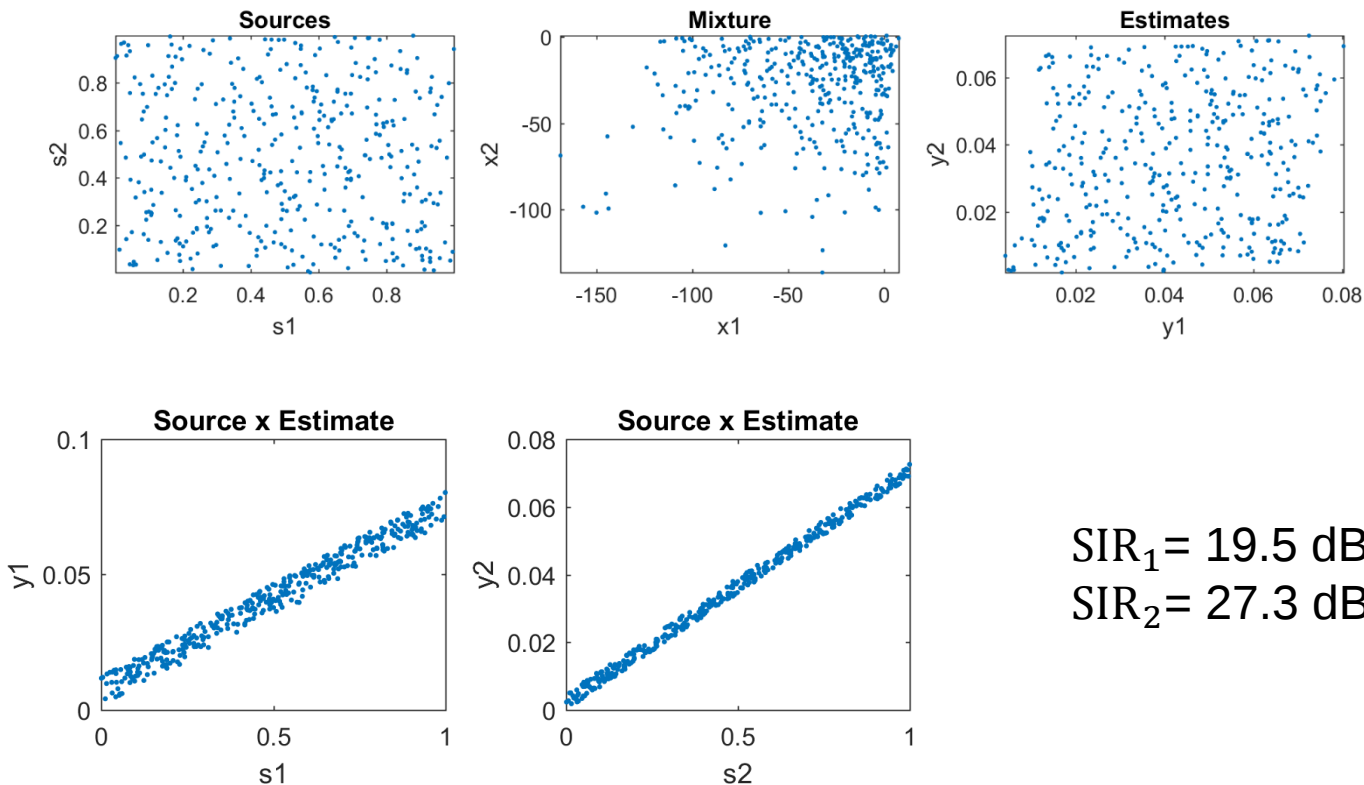
SIR values: 20.0 dB

26.6 dB

21.4 dB

Simulation: 2 sources with different valences

$z_1 = 1, z_2 = 2 \rightarrow 3^{\text{rd}}$ degree polynomial model



$SIR_1 = 19.5$ dB
 $SIR_2 = 27.3$ dB

Conclusion

- An extension to the algorithm proposed for a specific case has been presented and can be used for a broader range of source configurations.
- Computationally more expensive.
- Results not as accurate as obtained in the specific case, but the estimates are still reasonably good.
- Algorithm can be extended for any number of sources and valences.
- If polynomials of higher degree are used, possible problem with instability of recurrent network and/or ambiguity of solutions.
- Chemical data should probably not have such problems (mixing coefficients are low)

Thank you!

Questions?